

Mortality forecasting with the Lee-Carter method: Adjusting for smoothing and lifespan disparity

Ahbab Mohammad Fazle Rabbi,* Stefano Mazzuco†

Appendix A: Smoothing techniques and smoothing accuracy

Spline-based smoothing techniques

For one-dimensional smoothing, Hyndman and Ullah (2007) define log-mortality rates using the following smooth function with some observation error,

$$\log m_{x_i,t} = f_t(x_i^*) + \sigma_t(x_i^*)\epsilon_{t,i}. \quad (1)$$

Here, \log denotes the natural logarithm, $f_t(x)$ is the smooth function of age x_i with $i = 1, 2, \dots, p$ and $t = 1, 2, \dots, n$, x_i^* is the midpoint of age group x_i , and $\epsilon_{t,i}$ is an iid random variable which allows the amount of noise to vary with x_i . Assuming the life table deaths follow a Poisson distribution, Hyndman and Ullah (2007) estimated the variance of y_i as $\sigma^2 \approx 1/D_{x,t}$ by a Taylor series approximation. Here, $D_{x,t}$ is the number of deaths in year t of people aged x . To estimate $f_t(x)$, a nonparametric smoothing method can be used, such as splines.

Hyndman and Ullah (2007) applied weighted penalized regression splines to estimate $f_t(x)$ for each year. This weighting takes care of heterogeneity due to $\sigma_t(x)$ and is defined as being equal to the approximate inverse variances. Hyndman and Ullah (2007) preferred weighted penalized regression splines, as they offer shorter computational times. Moreover, these splines are constrained to ensure that the resulting function $f_t(x)$ is monotonically increasing for $x > c$ for some c (for example, 60 years). To implement this constraint, Hyndman and Ullah (2007) propose the use of a modified version of the method described by Wood (1994).

The second approach uses two-dimensional splines (Camarda 2012). This method fits a two-dimensional P-spline model with equally spaced B-splines along the X and Y axes (age and calendar year, respectively). With this approach, the response variables must comprise a matrix of Poisson-distributed death counts. For this, the splines offset can be provided; otherwise, all weights are assumed to be unity. In a Poisson regression setting applied to actual death counts, the offset will be the logarithm of the matrix of the exposure population.

In smoothing the mortality rates, this method utilizes a smoothing function (i.e. the Kronecker product of the B-spline basis over the two axes) and includes a discrete penalization directly on the differences of the B-spline coefficients. The smoothing parameters (designated by λ for each) are mainly used to tune the smoothness/accuracy of the fitted values (Currie et al. 2006). To optimize, the smoothing parameters, either the Akaike or Bayesian information criterion can be considered.

*fazlerabbi@stat.unipd.it

†mazzuco@stat.unipd.it

The accuracy measures of different smoothing techniques are summarized in Table 1–20. Comparison of smoothing techniques in terms of distribution of deaths for Swedish women are presented in Fig. 1. Errors of smoothing techniques for mortality rates of Swedish women are presented in Fig. 2.

Table 1: Accuracy of smoothing techniques for women of Australia (1950–2014).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	7.475	8.888	5.817
MSE(m_x) $\times 100$	1.524	2.246	1.068
ME(e_0)	-0.020	-0.003	0.005
MAE(e_0)	0.020	0.144	0.036

Table 2: Accuracy of smoothing techniques for women of Austria (1950–2014).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	9.100	10.177	8.862
MSE(m_x) $\times 100$	2.784	3.070	2.479
ME(e_0)	-0.025	0.001	0.027
MAE(e_0)	0.025	0.119	0.102

Table 3: Accuracy of smoothing techniques for women of Belgium (1950–2015).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	9.012	10.106	5.727
MSE(m_x) $\times 100$	2.582	2.874	1.453
ME(e_0)	-0.020	0.004	0.018
MAE(e_0)	0.020	0.132	0.031

Table 4: Accuracy of smoothing techniques for women of Canada (1950–2011).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	5.911	7.269	6.080
MSE(m_x) $\times 100$	0.985	1.402	0.926
ME(e_0)	-0.011	-0.001	0.017
MAE(e_0)	0.011	0.084	0.086

Table 5: Accuracy of smoothing techniques for women of Denmark (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	12.208	12.804	12.018
MSE(m_x) $\times 100$	4.680	4.528	4.194
ME(e_0)	-0.045	-0.002	0.010
MAE(e_0)	0.045	0.121	0.109

Table 6: Accuracy of smoothing techniques for women of Finland (1950–2015).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	12.367	13.062	12.310
MSE(m_x) $\times 100$	4.746	4.809	4.368
ME(e_0)	-0.048	-0.002	0.060
MAE(e_0)	0.048	0.127	0.138

Table 7: Accuracy of smoothing techniques for women of France (1950–2015).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	3.835	5.422	2.722
MSE(m_x) $\times 100$	0.523	0.908	0.314
ME(e_0)	-0.002	-0.001	-0.006
MAE(e_0)	0.003	0.145	0.016

Table 8: Accuracy of smoothing techniques for women of Germany (1956:2015).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	4.148	5.551	2.083
MSE(m_x) $\times 100$	0.573	0.955	0.189
ME(e_0)	-0.012	-0.015	0.007
MAE(e_0)	0.011	0.103	0.014

Table 9: Accuracy of smoothing techniques for women of Ireland (1950–2014).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	15.618	16.033	12.685
MSE(m_x) $\times 100$	6.114	5.876	4.211
ME(e_0)	-0.110	-0.006	0.078
MAE(e_0)	0.110	0.177	0.168

Table 10: Accuracy of smoothing techniques for women of Italy (1950–2014).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	4.260	6.072	3.910
MSE(m_x) $\times 100$	0.682	1.184	0.515
ME(e_0)	-0.003	-0.001	0.010
MAE(e_0)	0.003	0.169	0.086

Table 11: Accuracy of smoothing techniques for women of Japan (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	3.411	4.912	2.853
MSE(m_x) $\times 100$	0.601	0.897	0.425
ME(e_0)	-0.003	-0.005	0.005
MAE(e_0)	0.003	0.149	0.017

Table 12: Accuracy of smoothing techniques for women of Netherlands (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	6.220	7.560	3.712
MSE(m_x) $\times 100$	1.335	1.646	0.725
ME(e_0)	-0.010	-0.001	0.003
MAE(e_0)	0.011	0.102	0.047

Table 13: Accuracy of smoothing techniques for women of New Zealand (1950–2013).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	15.052	15.656	14.523
MSE(m_x) $\times 100$	5.881	5.742	4.981
ME(e_0)	-0.082	-0.004	0.001
MAE(e_0)	0.082	0.181	0.178

Table 14: Accuracy of smoothing techniques for women of Norway (1950–2014).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	11.801	12.976	11.541
MSE(m_x) $\times 100$	4.687	4.663	4.045
ME(e_0)	-0.040	-0.002	0.035
MAE(e_0)	0.040	0.106	0.078

Table 15: Accuracy of smoothing techniques for women of Portugal (1950–2015).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	7.908	9.029	5.400
MSE(m_x) $\times 100$	1.853	2.124	1.020
ME(e_0)	-0.024	0.008	0.035
MAE(e_0)	0.023	0.279	0.061

Table 16: Accuracy of smoothing techniques for women of Spain (1950–2014).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	7.095	7.990	5.523
MSE(m_x) $\times 100$	2.837	1.979	1.022
ME(e_0)	-0.018	-0.003	0.057
MAE(e_0)	0.018	0.203	0.167

Table 17: Accuracy of smoothing techniques for women of Sweden (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	10.408	11.281	10.716
MSE(m_x) $\times 100$	3.479	3.585	3.354
ME(e_0)	-0.029	-0.001	0.060
MAE(e_0)	0.029	0.097	0.102

Table 18: Accuracy of smoothing techniques for women of Switzerland (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	11.088	11.794	10.051
MSE(m_x) $\times 100$	4.102	4.028	3.269
ME(e_0)	-0.038	-0.002	0.015
MAE(e_0)	0.038	0.119	0.041

Table 19: Accuracy of smoothing techniques for women of the United Kingdom (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
MAE(m_x) $\times 100$	4.169	5.706	4.403
MSE(m_x) $\times 100$	0.531	0.948	0.544
ME(e_0)	-0.007	-0.006	0.006
MAE(e_0)	0.007	0.124	0.111

Table 20: Accuracy of smoothing techniques for women of the USA (1950–2016).

Measure of accuracy	1 dim Smoothing	2 dim Smoothing	<i>LASSO</i>
$\text{MAE}(m_x) \times 100$	3.170	4.817	1.909
$\text{MSE}(m_x) \times 100$	0.953	1.162	0.215
$\text{ME}(e_0)$	-0.154	0.139	0.950
$\text{MAE}(e_0)$	0.002	0.087	0.016

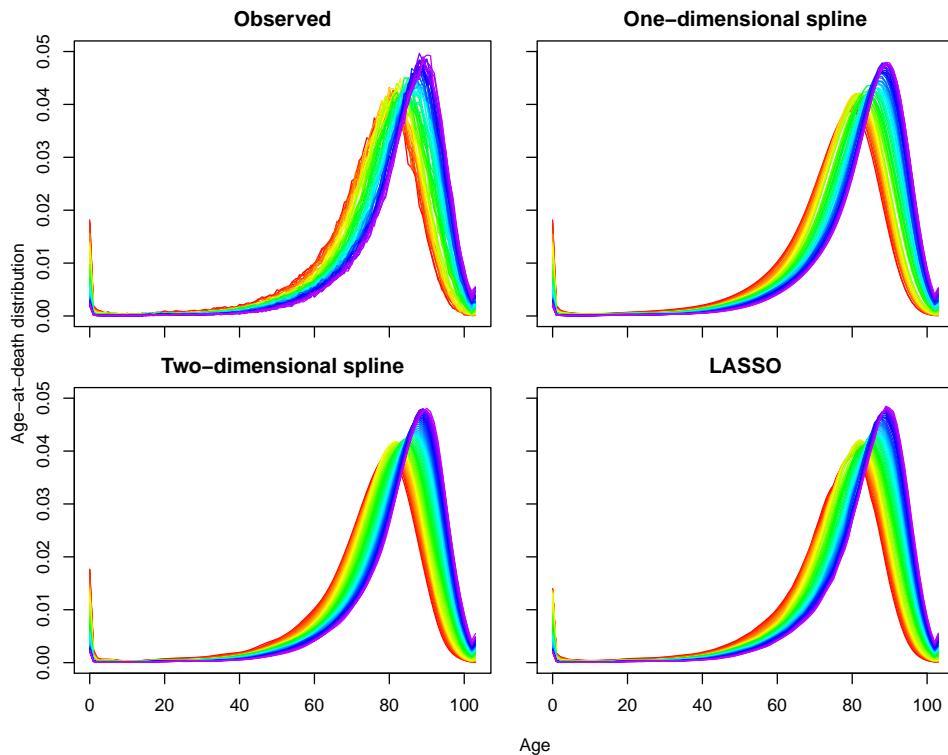


Fig. 1: Distribution of deaths from different smoothing techniques for Swedish women (1950–2016). Years are plotted using a rainbow palette, and so the earlier years are shown in red, followed by orange, yellow, green, blue, and indigo; the most recent years are plotted in violet.

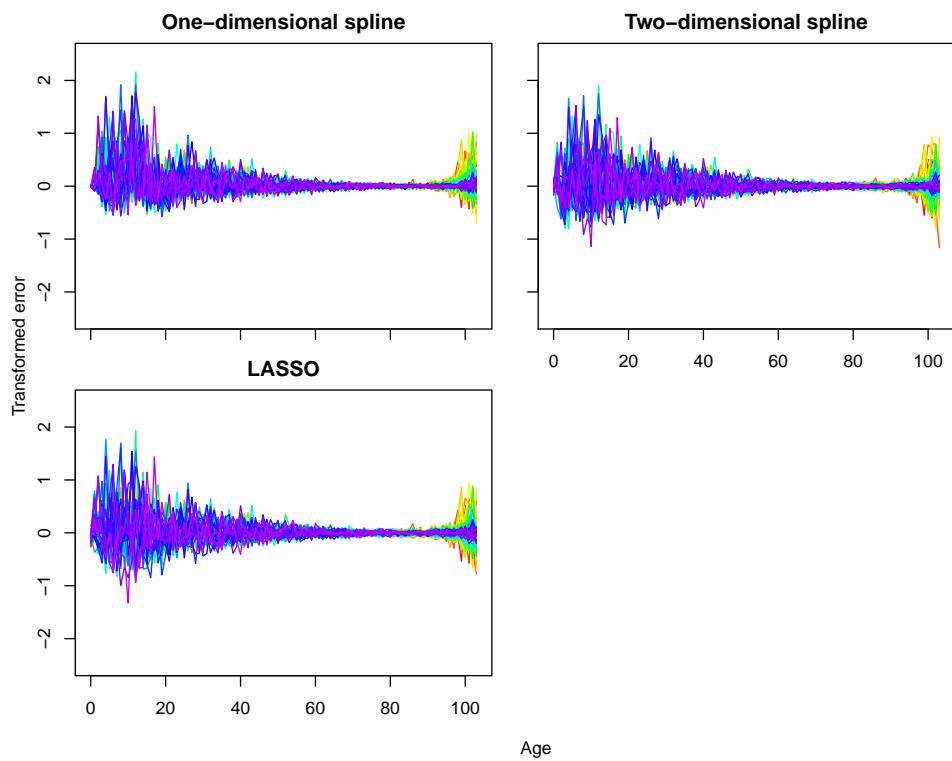


Fig. 2: Errors of smoothing techniques for female mortality of Sweden (1950–2016). Years are plotted using a rainbow palette, as before.

Appendix B: Assessment of the forecast techniques

Effect of smoothing and new adjustment policy in model-fitting are presented in Fig. 3. Relation between life expectancy and lifespan disparity are summarized in Table 21. Forecast of mortality rates for Swedish women are presented in Fig. 4. Variance explained by the models are summarized in Table 22 and other measures of forecast accuracies are presented in Table 23–27.

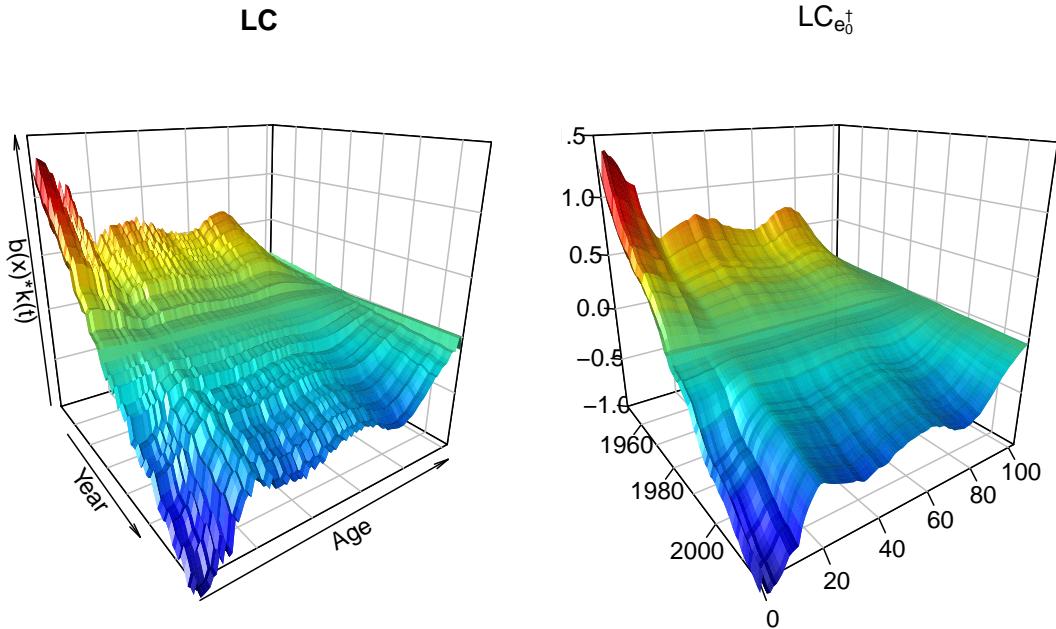


Fig. 3: Product of the age component and time component for Swedish women (1950–2016) using LC (Lee and Carter 1992) and the proposed $LC_{e_0^{\dagger}}$.

Table 21: Relation between life expectancy at birth (e_0) and lifespan disparity (e_0^{\dagger}) for Denmark, Japan, Sweden, and the USA. Values are presented in terms of Pearson correlation coefficient (r).

Country	Method or source		
	LC	HMD	$LC_{e_0^{\dagger}}$
Denmark	-0.993	-0.941	-0.994
Japan	-0.998	-0.937	-0.997
Sweden	-0.999	-0.987	-0.999
USA	-0.999	-0.981	-0.999

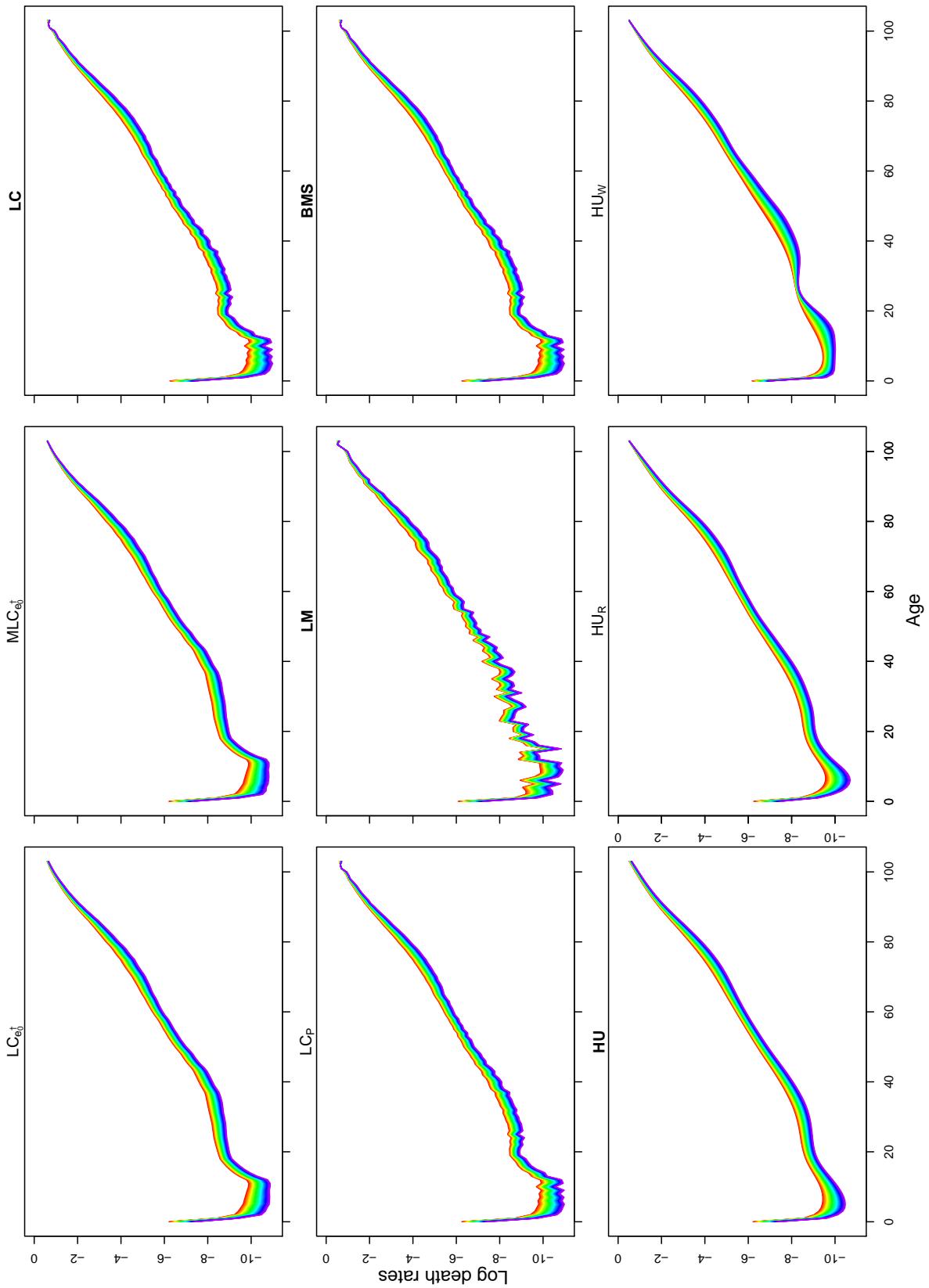


Fig. 4: Mortality forecast for Swedish women by different methods (2017–2050). Years are plotted using a rainbow palette, as before.

Appendix C: R-codes for the proposed model

```
library(demography)
library(smoothAPC)
library(forecast)
library(ftsa)

data #demogdata object for observed mortality rates#

years <- data$year

#estimation of the observed lifespan disparity#
ltjd<-print(lifetable(data, series="female",
max.age = max(data$age)),type = c("period"))

eddgg<-sapply(ltjd, FUN = function(x) sum(x$ex*x$lx*x$mx))

overall_edgr <- cbind.data.frame("years"=years,"edagger"=eddgg)
plot(overall_edgr)

#estimation of fitted lifespan disparity#
#basic Lee-Carter model is fitted on smoothed mortality data#
# ddata1 is the demogdata object containing pre-smoothed #
# mortality rates using LASSO#

ln.female <- lca(ddata1, max.age= max(ddata1$age),
adjust = "none", interpolate=TRUE)
a1j<-ln.female$ax
b1j<-ln.female$bx
k1j<-ln.female$kt

# for maximum likelihood approach, these initial parameters are
# obtained using StMoMo package and kept constant in the main model

#library(StMoMo)

#data.f<-StMoMoData(ddata1, series = "female", type = "central")

#LC<-lc(link = "log", const = "sum")
#LC<-lc()

#ages.fit<-ages1
#years.fit<-years1

#LCfit<-fit(object = LC,data = data.f, ages.fit = #ages.fit,years.fit = years.fit)

#a1j<-LCfit$ax
#b1j<-as.vector(LCfit$bx)
```

```

#k1j<-as.vector(LCfit$kt)

length(a1j)
length(b1j)
length(k1j)

mx1j<-exp(b1j*k1j[1]+a1j)

# for getting fitted mortality rates from estimated parameters #
# of LC model#
fitmx <- function (kt,ax,bx,transform=FALSE)
{
clogratesfit <- outer(kt, bx)
logratesfit <- sweep(clogratesfit,2,ax,"+")
if(transform)
return(logratesfit)
else
return(exp(logratesfit))
}

wwmxj<-fitmx(k1j,a1j,b1j)
wmxj<-t(wwmxj)
plot(wmxj)

ages1 <- ddata1$age
years1 <- ddata1$year
fDx1 <- matrix(wmxj,
ncol = length(years1),
nrow = length(ages1))
fNx1 <- usa$pop$female

lcfdata1 <- demogdata(data = fDx1, pop = fNx1, ages = ages1,
years = years1, type = "mortality", label = "label", name = "female")

plot(lcfdata1)

#extracting component of e-dagger from fitted LC to use later #
fitswed <- print(lifetable(lcfdata1,years = ddata1$year,
ages = ages1,max.age = max(ddata1$age)),type = c("period"))

years<-1950:2016

fitted_edgr <- sapply(fitswed, FUN = function(x) sum(x$ex*x$lx*x$mx))

#extracting lx, ex from life table obtained from fitted LC #
llxx <- matrix(unlist(lapply(fitswed, FUN = function(x) x[, "lx"])),
nrow = length(years), ncol = length(ages1), byrow = TRUE)

```

```

eexx <- matrix(unlist(lapply(fitswed, FUN = function(x) x[, "ex"])),  

nrow = length(years), ncol = length(ages1), byrow = TRUE)

mmxx <- matrix(unlist(lapply(fitswed, FUN = function(x) x[, "mx"])),  

nrow = length(years), ncol = length(ages1), byrow = TRUE)

#the function for lc_edagger#

lcadagger<-function (data, series = names(data$rate)[1],  

years = data$year, ages = data$age, max.age = max(data$age),  

adjust = c("edagger", "none"), chooseperiod = FALSE, minperiod = 20,  

breakmethod = c("bai",  

"bms"), scale = FALSE, restype = c("logrates", "rates",  

"deaths"), interpolate = TRUE)
{
  if (class(data) != "demogdata") {
    stop("Not demography data")
  }
  if (!any(data$type == c("mortality", "fertility"))) {
    stop("Neither mortality nor fertility data")
  }
  is.el <- function(el, set)
  {
    is.element(toupper(el), toupper(set))
  }

  # Compute expected age from single year mortality rates
  get.e0 <- function(x, agegroup, sex, startage=0)
  {
    lt(x, startage, agegroup, sex)$ex[1]
  }

  # Replace zeros with interpolated values
  fill.zero <- function(x, method="constant")
  {
    tt <- 1:length(x)
    zeros <- abs(x) < 1e-9
    xx <- x[!zeros]
    tt <- tt[!zeros]
    x <- stats::approx(tt, xx, 1:length(x), method=method, f=0.5, rule=2)
    return(x$y)
  }

  adjust <- match.arg(adjust)
  restype <- match.arg(restype)
  breakmethod <- match.arg(breakmethod)
  data <- extract.ages(data, ages, combine.upper = FALSE)
  if (max.age < max(ages))

```

```

data <- extract.ages(data, min(ages):max.age, combine.upper = TRUE)
startage <- min(data$age)
get.series <- function(data,series)
{
  if(!is.el(series,names(data)))
    stop(paste("Series",series,"not found"))
  i <- match(toupper(series),toupper(names(data)))
  return(as.matrix(data[[i]]))
}

mx <- get.series(data$rate, series)
pop <- get.series(data$pop, series)
startyear <- min(years)
stopyear <- max(years)
if (startyear > max(data$year) | stopyear < min(data$year))
  stop("Year not found")
startyear <- max(startyear, min(data$year))
if (!is.null(stopyear))
  stopyear <- min(stopyear, max(data$year))
else stopyear <- max(data$year)
id2 <- stats::na.omit(match(startyear:stopyear, data$year))
mx <- mx[, id2]
pop <- pop[, id2]
year <- data$year[id2]
deltat <- year[2] - year[1]
ages <- data$age
n <- length(ages)
m <- sum(id2 > 0)
edgr<-overall_edgr$edagger
mx <- matrix(mx, nrow = n, ncol = m)
if (interpolate) {
  mx[is.na(mx)] <- 0
  if (sum(abs(mx) < 1e-09, na.rm = TRUE) > 0) {
    warning("Replacing zero values with estimates")
    for (i in 1:n) mx[i, ] <- fill.zero(mx[i, ])
  }
}
mx <- t(mx)
mx[mx == 0] <- NA
logrates <- log(mx)
pop <- t(pop)
deaths <- pop * mx
ax <- apply(logrates, 2, mean, na.rm = TRUE)
if (sum(ax < -1e+09) > 0)
  stop(sprintf("Some %s rates are zero.\n Try reducing the maximum age
or setting interpolate=TRUE.", data$type))
clogrates <- sweep(logrates, 2, ax)
svd.mx <- svd(clogrates)
sumv <- sum(svd.mx$v[, 1])
bx <- svd.mx$v[, 1]/sumv
kt <- svd.mx$d[1] * svd.mx$u[, 1] * sumv

```

```

ktadj <- rep(0, m)
logdeathsadj <- matrix(NA, n, m)
z <- log(t(pop)) + ax
x <- 1:m
ktse <- stats::predict(stats::lm(kt ~ x), se.fit = TRUE)$se.fit
ktse[is.na(ktse)] <- 1
agegroup = ages[4] - ages[3]

edgr<-overall_edgr$edagger

fitmx <- function (kt,ax,bx,transform=FALSE)
{
# Derives mortality rates from kt mortality index,
# per Lee-Carter method
clogratesfit <- outer(kt, bx)
logratesfit <- sweep(clogratesfit,2,ax,"+")
if(transform)
return(logratesfit)
else
return(exp(logratesfit))
}
#to obtaining root, following function from demography package is used#
findroot <- function(FUN,guess,margin,try=1,...)
{
# First try in successively larger intervals around best guess
for(i in 1:5)
{
rooti <- try(stats::uniroot(FUN,
interval=guess+i*margin/3*c(-1,1),...),silent=TRUE)
if(class(rooti) != "try-error")
return(rooti$root)
}
# No luck. Try really big intervals
rooti <- try(stats::uniroot(FUN,
interval=guess+10*margin*c(-1,1),...),silent=TRUE)
if(class(rooti) != "try-error")
return(rooti$root)

# Still no luck. Try guessing root using quadratic approximation
if(try<3)
{
root <- try(quadroot(FUN,guess,10*margin,...),silent=TRUE)
if(class(root)!="try-error")
return(findroot(FUN,root,margin,try+1,...))
root <- try(quadroot(FUN,guess,20*margin,...),silent=TRUE)
if(class(root)!="try-error")
return(findroot(FUN,root,margin,try+1,...))
}

# Finally try optimization
root <- try(newroot(FUN,guess,...),silent=TRUE)

```

```

if(class(root)!="try-error")
return(root)
else
root <- try(newroot(FUN,guess-margin,...),silent=TRUE)
if(class(root)!="try-error")
return(root)
else
root <- try(newroot(FUN,guess+margin,...),silent=TRUE)
if(class(root)!="try-error")
return(root)
else
stop("Unable to find root")
}

quadroot <- function(FUN,guess,margin,...)
{
x1 <- guess-margin
x2 <- guess+margin
y1 <- FUN(x1,...)
y2 <- FUN(x2,...)
y0 <- FUN(guess,...)
if(is.na(y1) | is.na(y2) | is.na(y0))
stop("Function not defined on interval")
b <- 0.5*(y2-y1)/margin
a <- (0.5*(y1+y2)-y0)/(margin^2)
tmp <- b^2 - 4*a*y0
if(tmp < 0)
stop("No real root")
tmp <- sqrt(tmp)
r1 <- 0.5*(tmp-b)/a
r2 <- 0.5*(-tmp-b)/a
if(abs(r1) < abs(r2))
root <- guess+r1
else
root <- guess+r2
return(root)
}

# Try finding root using minimization
newroot <- function(FUN,guess,...)
{
fred <- function(x,...){(FUN(x,...)^2)}
junk <- stats::nlm(fred,guess,...)
if(abs(junk$minimum)/fred(guess,...) > 1e-6)
warning("No root exists. Returning closest")
return(junk$estimate)
}
if (adjust == "edagger") {

Fundg<-function(p,bx,ax,edgr,llxxi,eexxi){

```

```

edgr = sum(exp(ax + bx*p)*llxxi*eexxi)
}
for (i in 1:m) {
  if (i == 1)
    guess <- kt[1]
  else guess <- mean(c(ktadj[i - 1], kt[i]))
  ktadj[i] <- findroot(Fundg, guess = guess, margin = 3 *
  ktse[i], edgr=edgr[i] , llxxi=llxx[i,],
  eexxi=eexx[i,],ax = ax, bx = bx)
  logdeathsadj[,i]<-z[,i]+bx*ktadj[i]
}
}
else if (adjust == "none")
  ktadj <- kt
else stop("Unknown adjustment method")
kt <- ktadj

#the following part is needed in case of choosing best fitting#
#period#
if (chooseperiod) {
  if (breakmethod == "bai") {
    x <- 1:m
    bp <- strucchange::breakpoints(kt ~ x)$breakpoints
    bp <- bp[bp <= (m - minperiod)]
    bestbreak <- max(bp)
    return(lca(data, series, year[(bestbreak + 1):m],
    ages = ages, max.age = max.age, adjust = adjust,
    interpolate = interpolate, chooseperiod = FALSE,
    scale = scale))
  }
  else {
    RS <- devlin <- devadd <- numeric(m - 2)
    for (i in 1:(m - 2)) {
      tmp <- lcadagger(data, series, year[i:m], ages = ages,
      max.age = max.age, adjust = adjust, chooseperiod = FALSE,
      interpolate = interpolate, scale = scale)
      devlin[i] <- tmp$mdev[2]
      devadd[i] <- tmp$mdev[1]
      RS[i] <- (tmp$mdev[2]/tmp$mdev[1])
    }
    bestbreak <- order(RS[1:(m - minperiod)])[1] - 1
    out <- lcadagger(data, series, year[(bestbreak + 1):m],
    ages = ages, max.age = max.age, adjust = adjust,
    chooseperiod = FALSE, interpolate = interpolate,
    scale = scale)
    out$mdevs <- ts(cbind(devlin, devadd, RS), start = startyear,
    deltat = deltat)
    dimnames(out$mdevs)[[2]] <- c("Mean deviance total",
    "Mean deviance base", "Mean deviance ratio")
    return(out)
  }
}

```

```

}

logfit <- fitmx(kt, ax, bx, transform = TRUE)
if (restype == "logrates") {
  fit <- logfit
  res <- logrates - fit
}
else if (restype == "rates") {
  fit <- exp(logfit)
  res <- exp(logrates) - fit
}
else if (restype == "deaths") {
  fit <- exp(logfit) * pop
  res <- deaths - fit
}
residuals <- fts(ages, t(res), frequency = 1/deltat,
start = years[1], xname = "Age", yname = paste("Residuals",
data$type,
"rate"))
fitted <- fts(ages, t(fit), frequency = 1/deltat, start = years[1],
xname = "Age", yname = paste("Fitted", data$type, "rate"))
names(ax) <- names(bx) <- ages
if (scale) {
  avdiffk <- -mean(diff(kt))
  bx <- bx * avdiffk
  kt <- kt/avdiffk
}
deathsadjfit <- exp(logfit) * pop
drift <- mean(diff(kt))
ktlinfit <- mean(kt) + drift * (1:m - (m + 1)/2)
deathslinfit <- fitmx(ktlinfit, ax, bx, transform = FALSE) *
pop
dflogadd <- (m - 2) * (n - 1)
mdevlogadd <- 2/dflogadd * sum(deaths * log(deaths/deathsadjfit) -
(deaths - deathsadjfit))
dfloglin <- (m - 2) * n
mdevloglin <- 2/dfloglin * sum(deaths * log(deaths/deathslinfit) -
(deaths - deathslinfit))
mdev <- c(mdevlogadd, mdevloglin)
output <- list(label = data$label, age = ages, year = year,
mx = t(mx), ax = ax, bx = bx, kt = ts(kt, start = startyear,
deltat = deltat), residuals = residuals, fitted = fitted,
varprop = svd.mx$d[1]^2/sum(svd.mx$d^2), y = fts(ages,
t(mx), start = years[1], frequency = 1/deltat, xname = "Age",
yname = ifelse(data$type == "mortality", "Mortality",
"Fertility")), mdev = mdev)
names(output)[4] <- series
output$call <- match.call()
names(output$mdev) <- c("Mean deviance base", "Mean deviance total")
output$adjust <- adjust
output$type <- data$type
return(structure(output, class = "lca"))

```

```
}
```

```
#to fit the model#  
  
funmod2 <- lcadagger(ddata1, ages=ddata1$age,  
max.age = max(ddata1$age), adjust = "edagger", interpolate=TRUE)  
  
# to see the fitted parameters, variation explained#  
funmod2  
  
#for forecast t years ahead mortality forecast#  
forecast.female <- forecast(funmod2, h=t, jumpchoice = "actual")  
plot(forecast.female, main=expression(LC[e[0]^"\u2020"]))  
  
# t years ahead 95% prediction interval for life expectancy #  
# at birth #  
fun2<- e0(forecast(funmod2, level=95, max.age=max(ddata1$age),  
jumpchoice = "actual", h=t), PI=TRUE)  
  
plot(fun2, main=expression(LC[e[0]^"\u2020"]))
```

References

- Camarda, C. G. (2012). MortalitySmooth: An R package for smoothing poisson counts with P-splines. *Journal of Statistical Software*, 50(1):1–24.
- Currie, I. D., Durban, M., and Eilers, P. H. (2006). Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(2):259–280.
- Hyndman, R. J. and Ullah, M. S. (2007). Robust forecasting of mortality and fertility rates: a functional data approach. *Computational Statistics & Data Analysis*, 51(10):4942–4956.
- Lee, R. D. and Carter, L. R. (1992). Modeling and forecasting us mortality. *Journal of the American statistical association*, 87(419):659–671.
- Wood, S. N. (1994). Monotonic smoothing splines fitted by cross validation. *SIAM Journal on Scientific Computing*, 15(5):1126–1133.